

# Assessment

Marking will be via completion of a programming project of your own choosing. Ideally it should be something relevant to your study or research, but a small list of ideas are provided below. Some guidelines:

- It should be non-trivial.
- It should be complete. That is, while it can be written so as to be extended or incorporated into a larger project *later*, what is presented must be functional *now*. It can be a prototype, but it has to work.
- Extensions or add-ons to other frameworks (e.g. BioPython) are acceptable, as long as their use is demonstrated.
- The resultant software doesn't have to be pretty, but it should be usable and reusable. As a rule of thumb, a script that will only be run once isn't acceptable.
- Given the time constraints of the course, you will probably have to work out some material before it is covered in class. For example, if you're using a web framework, be aware we may not cover them until late in the course.
- All project proposals have to be vetoed by myself and should be fixed by mid week 3. If need be, project aims can be renegotiated along the way. Presentations will be given on progress in weeks 4 and 5 and the marking based on a final presentation and examination of the code in week 6. Unfortunately no extensions are possible.
- You can work in pairs if need be.

## Project suggestions

- A web service storing, visualising and search some type of biological data. *Turbogears and Django are probably the web toolkits that are quickest to get up to speed on.*)
- (For Java programmers only.) Java has some useful libraries and features that Python doesn't (e.g JDBC). Fortunately, there are ways to wrap Java or call in from Python. Make one of these services accessible from Python.
- Many types of biological data are graph-like in nature (e.g metabolic networks, gene networks). Build a framework for visualising and manipulating these in Python. *It may make sense to build this on top of an existing framework, like NetworkX.*
- BioPython sucks. Find an item on BioPython's bug list, or part of it that is poorly designed and fix it. *CoreBio could use some of BioPython's functions so design it for inclusion in that library.*
- Design a module for representing phylogenies, rooted and unrooted, with attendant functions. *Of course, any library should be documented, tested and demonstrated.*

- Design a module for drawing phylogenies. *Maybe you can work with someone on the previous project. The SPNG module may be a good drawing toolkit.*
- Write a GUI or parser for creating and editing one of the many complicated biological data formats: NEXUS, BioEdit etc.